Basic Statistic:

$Var = \frac{1}{n-1}\sum(X_i - \bar{X})^2$

$\sigma_X = \sqrt{Var}$

$Cor = \frac{\sum(X_i-\bar{X})(Y_i-\bar{Y})}{\sqrt{\sum(X_i-\bar{X})^2\sum(Y_i-\bar{Y})^2}}$

$Cor = \frac{\sum(X_i-\bar{X})(Y_i-\bar{Y})}{(n-1)\sigma_X\sigma_Y}$

$Standardize = \frac{X_i-\bar{X}}{\sigma_X}$

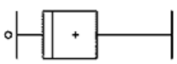
$Normalize = \frac{X_i-Min(X)}{Max(X)-Min(X)}$

Visualization:

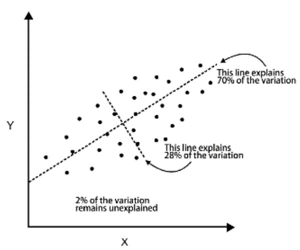
Boxplots:

$Max = Q3 + 1.5(Q3 - Q1)$

$Min = Q1 - 1.5(Q3 - Q1)$



Principal Components Analysis:



$Z_j(i) = \sum_{k=1}^{p} a_{jk}(X_k(i) - \bar{X}_k) \quad \bar{Z}_j = 0$

$\sum_{j=1}^{p} Var[X_j] = \sum_{j=1}^{p} Var[Z_j]$

$Cov\ matrix = \begin{bmatrix} Var[X] & Cov[X,Y] \\ Cov[X,Y] & Var[Y] \end{bmatrix}$

$Cov\ matrix = \begin{bmatrix} Var[Z_1] & 0 \\ 0 & Var[Z_2] \end{bmatrix}$

$Var[Z_1] \geq Var[Z_2] \geq \cdots \geq Var[Z_p]$

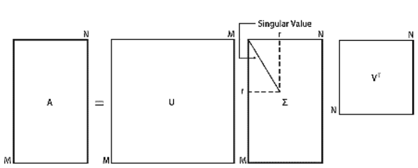
$Var\% = \frac{\sum_{j=1}^{m} Var[Z_j]}{\sum_{j=1}^{p} Var[Z_j]} = \frac{\sum_{j=1}^{m} Var[Z_j]}{\sum_{j=1}^{p} Var[X_j]}$

Disadvantages:

Lose predictive information that is nonlinear

Singular Value Decomposition:

$\boldsymbol{A = U\Sigma V^T} \quad \boldsymbol{(A^T A)v_i} = \lambda_i \boldsymbol{v_i} \quad \boldsymbol{(AA^T)u_i} = \lambda_i \boldsymbol{u_i}$



Numerical Performance Measure:

Error/ Residual: $e_i = y_i - \hat{y}_i$

Mean Absolute Error/Deviation:

$MAE\ or\ MAD = \frac{1}{n}\sum_{i=1}^{n}|e_i|$

Average Error:

$AE = \frac{1}{n}\sum_{i=1}^{n} e_i$

Mean Absolute Percentage Error:

$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{e_i}{y_i}\right| \times 100\%$

Root-Mean-Squared Error:

$RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n} {e_i}^2}$

Error Sum of Squares:

$SSE = \sum(y_i - \hat{y}_i)^2$

Regression Sum of Squares:

$SSR = \sum(\hat{y}_i - \bar{y}_i)^2$

Total Sum of Squares:

$SST = \sum(y_i - \bar{y})^2$

$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$

${R^2}_{adj} = 1 - \frac{n-1}{n-p-1}(1 - R^2)$

Classification Performance Measure:

Benchmark: The Naïve Rule 50% when 1:1

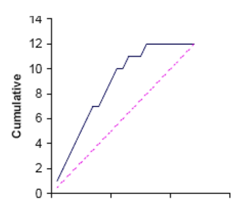
Lift Chart:

① Ordering the set of records (validation data) by their predicted value in descending order

② Compute cumulative actual value from <u>top</u> to <u>bottom</u>: $C(i)$

③ Compute cumulative average actual value from top to bottom: $A(i)$

④ Plot C(i) versus A(i) versus on the same graph



Adding Costs/Benefits to Lift Curve:

Y-axis is cumulative cost/benefit

Line may have negative slope

Decile-wise Lift Chart:

Order → Group into 10 Deciles → Compute Decile Mean → Compute Global Mean → Ratio → Plot Histograms

Confusion Matrix:

	Predicted Positive Class	Predicted Negative Class
Actual Positive Class	$TP = n_{11}$	$FN = n_{12}$
Actual Negative Class	$FP = n_{21}$	$TN = n_{22}$

$Error\ Rate = \frac{FP + FN}{TP + FP + FN + TN}$

$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$

$Sensitivity = \frac{TP}{TP + FN}$

$Specificity = \frac{TN}{FP+TN}$

$1 - Specificity = \frac{FP}{FP+TN}$

$False\ Postive\ Rate = \frac{FP}{TP+FP}$

$False\ Negative\ Rate = \frac{FN}{FN+TN}$

$Positive\ Predicitve\ Value = \frac{TP}{TP+FP}$

$Negative\ Predicitve\ Value = \frac{TN}{FN + TN}$

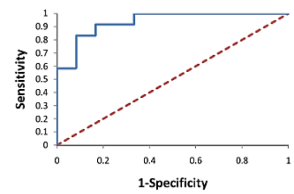
$Precision = Positive\ Predicitve\ Value$

$Recall = Sensitivity$

$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$

$F - Measure = \frac{2TP}{2TP + FP + FN}$

ROC Curve:



Compute (Sensitivity, 1-Specificity) for by varying cutoff from 0 to 1 and plot

Asymmetric Costs:

$q_1 = cost\ of\ misclassifying\ an\ actual\ C_1$

$q_2 = cost\ of\ misclassifying\ an\ actual\ C_2$

$p_1 = proportion\ of\ C_1$

$p_2 = proportion\ of\ C_2$

$Average\ Cost = (1 - Sensitivity)p_1q_1 + (1 - Specificity)p_2q_2$

$Average\ Cost = p_1q_1 \times \left[(1 - Sensitivity) + (1 - Specificity)\frac{p_2\,q_2}{p_1\,q_1}\right]$

Oversampling:

Train the model on oversampled data (1:1) but validate it with regular data.

Adjusting the Confusion Matrix:

$f_i = \frac{Proportion\ of\ C_i\ in\ Oversample\ Data}{Proportion\ of\ C_i\ in\ Original\ Data}$

$n_{ij} = \frac{m_{ij}}{f_i}$

Multiple Linear Regression:

Perceptron Learning Algorithm (PLA):

$y_n = \pm 1$

Introduce $x_0 = 1$, $h(\boldsymbol{x}) = sign(\boldsymbol{\beta^T x})$

① Pick a misclassified point: $h(\boldsymbol{x_i}) \neq y_i$

② Update: $\boldsymbol{\beta} \coloneqq \boldsymbol{\beta} + y_i \boldsymbol{x_i}$

Hypothesis:

$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \varepsilon$

$h(\boldsymbol{x}) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p = \boldsymbol{\beta}^{\boldsymbol{T}} \boldsymbol{x}\ (x_0 = 1)$

Dummy: $\#\ category - 1$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix} \quad \boldsymbol{x}^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_p^{(i)} \end{bmatrix} \quad \boldsymbol{X} = \begin{bmatrix} \left(\boldsymbol{x}^{(1)}\right)^{\boldsymbol{T}} \\ \left(\boldsymbol{x}^{(2)}\right)^{\boldsymbol{T}} \\ \vdots \\ \left(\boldsymbol{x}^{(m)}\right)^{\boldsymbol{T}} \end{bmatrix} \quad \boldsymbol{Y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

Cost Function:

$$J(\boldsymbol{\beta}) = \frac{1}{2m} \sum_{i=1}^{m} \left(\boldsymbol{\beta}^{\boldsymbol{T}} \boldsymbol{x}^{(i)} - y^{(i)}\right)^2$$

① Gradient Descent:

$$\beta_j \coloneqq \beta_j - \alpha \frac{\partial J(\boldsymbol{\beta})}{\partial \beta_j}$$

for i = 1 to m

$$\beta_j \coloneqq \beta_j - \frac{\alpha}{m} \left(\boldsymbol{\beta}^{\boldsymbol{T}} \boldsymbol{x}^{(i)} - y^{(i)}\right) x_j^{(i)}$$

$$\boldsymbol{\beta} \coloneqq \boldsymbol{\beta} - \frac{\alpha}{m} \boldsymbol{X}^{\boldsymbol{T}} (\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{Y})$$

repeat until convergence

② Normal Equations:

$\boldsymbol{\beta} = (\boldsymbol{X}^{\boldsymbol{T}} \boldsymbol{X})^{-1} \boldsymbol{X}^{\boldsymbol{T}} \boldsymbol{Y}$

$\boldsymbol{X}^{\boldsymbol{T}} \boldsymbol{X}$ should be invertible

Assumptions:

① Normality: ε follows a normal distribution

② Linearity: linear relationship

③ Independence: Y_i is independent

④ Homoscedasticity: constant variance of ε

Reports for Training & Validation:

①SSE ②RMS ③AE

Performance Improvement:

① Add polynomial: x_j^k

② Add interaction effects: $x_k x_j$

③ Convert numerical to a binary: $x_j > a$

Subset Selection:

One rough rule of thumb: $m > 5(p+2)$

Methods:

① Exhaustive Search

② Partial Search Algorithms:

Forward Selection: No predictors → Add one by one → Stop when not significant (missing pairs)

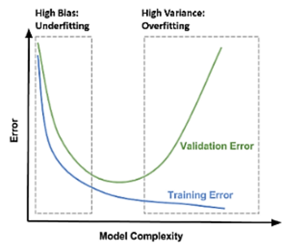
Backward Elimination: All predictors → eliminate one by one → Stop when all significant (time consuming and unstable)

Stepwise Regression: Like Forward Selection, also drop non-significant predictors

Criterion: high $R^2{}_{adj}$, $C_p \approx p + 1$, small p

$$C_p = \frac{SSE}{\hat{\sigma}_{Full}{}^2} + 2(p+1) - n$$

Bias-Variance Trade off:



Learning Curve:

Regularization:

$J(\boldsymbol{\beta}) \mathrel{+}= \frac{\lambda}{2m} \sum_{j=1}^{p} \beta_j{}^2 \qquad \frac{\partial J(\boldsymbol{\beta})}{\partial \beta_j} \mathrel{+}= \frac{\lambda}{m} \beta_j$

Strategies:

① Fix high bias:

Adding features, Polynomial, Decreasing λ, More nodes and hidden layers

② Fix high variance:

Get more training samples, Smaller set of features, Increase λ, Less nodes and hidden layers

Distance Measure:

① Euclidean Distance:

$$\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + \left(x_p - y_p\right)^2}$$

Drawback: Sensitive to scale and variance, ignores correlation

② Manhattan Distance:

$|x_1 - y_1| + |x_2 - y_2| + \cdots + \left|x_p - y_p\right|$

③ Statistical (Mahalanobis) Distance:

$[\boldsymbol{X} - \boldsymbol{Y}]^{\boldsymbol{T}} \boldsymbol{S}^{-1} [\boldsymbol{X} - \boldsymbol{Y}]$

$[\boldsymbol{X} - \boldsymbol{Y}]^{\boldsymbol{T}} = [x_1 - y_1 \quad x_2 - y_2 \quad \cdots \quad x_p - y_p]$

$\boldsymbol{S} = cov(\boldsymbol{X}, \boldsymbol{Y}) =$

$$\begin{bmatrix} (x_1 - y_1)(x_1 - y_1) & (x_1 - y_1)(x_2 - y_2) & \dots & (x_1 - y_1)(x_p - y_p) \\ (x_2 - y_2)(x_1 - y_1) & (x_2 - y_2)(x_2 - y_2) & \dots & (x_2 - y_2)(x_p - y_p) \\ \vdots & \vdots & \vdots & \vdots \\ (x_p - y_p)(x_1 - y_1) & (x_p - y_p)(x_2 - y_2) & \dots & (x_p - y_p)(x_p - y_p) \end{bmatrix}$$

④ Maximum Coordinate Distance:

$\max_i |x_i - y_i|$

Nonnegative: $d_{ij} \geq 0$

Self-proximity: $d_{ii} = 0$

Symmetry: $d_{ij} = d_{ji}$

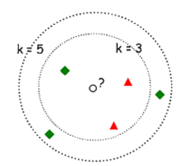
Triangle inequality: $d_{ij} \leq d_{ik} + d_{kj}$

k-Nearest Neighbors:

Find the nearest k neighbors

Use a majority decision rule

Categorical should be converted into dummy



Characteristics:

Classification, Data-driven, No assumption about the data, Curse of dimensionality

Low k vs. High k:

① Low k: Capture local structure, May fit the noise

② High k: May miss local structure, More smoothing, less noise

Cutoff Value:

① Simple majority rule: 0.5

② Choose other cutoff value to maximize accuracy or incorporate misclassification costs.

Numerical Prediction:

Average of response values, may be weighted

Naïve Bayes:

Characteristics:

Used only with categorical predictors (numerical must be binned to categorical), Classification, Data-driven, No assumption about the data, Can be used for large data

Exact Bayes Classifier:

$P\left(C_k | \boldsymbol{x}^{(i)}\right) = \frac{P\left(x^{(i)} | C_k\right) P(C_k)}{\sum_{i=1}^{c} P\left(x^{(i)} | C_i\right) P(C_i)}$

With large data sets, may be hard to find other records that exactly match the record

Assumption of independence:

$P(x_1, \dots x_p | C_k) = \prod_{i=1}^{p} P(x_i|C_k)$

Naïve Bayes Classifier:

$P(C_k|x_1, \dots x_p) = \frac{[\prod_{i=1}^{p} P(x_i|C_k)]P(C_k)}{\sum_{j=1}^{c}[\prod_{i=1}^{p} P(x_i|C_j)]P(C_j)}$

Laplacian Smoothing:

when a predictor category is not present in training data, joint probability becomes 0

$P(x_i|C_k) = \frac{\#x_i \& C_k + l}{\#C_k + l|x_i|}$

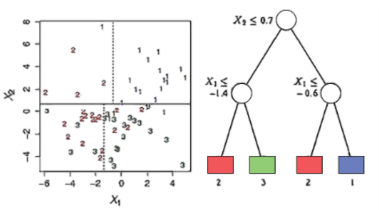
l is the Laplace smoothing factor

$|x_i|$ is # values x_i can take on (# category)

Classification and Regression Trees:

Trees are based on separating observations into subgroups by creating splits on predictors

Can work with missing data



Characteristics:

Classification, Data-driven

Impurity Measure:

Gini Index: $I(A) = 1 - \sum_{i=1}^{c} {p_i}^2$

Entropy: $Entropy(A) = -\sum_{i=1}^{c} p_i log_2\, p_i$

Min: $I(A) = 0$, $Entropy(A) = 0$,

when all cases belong to same class (most pure)

Max: $I(A) = 1 - \frac{1}{c}$, $Entropy(A) = log_2 c$,

when all classes are equally represented

Combined Impurity: Weighted Sum of Impurity

Impurity and Recursive Partitioning:

① Obtain overall impurity measure

② compare this measure across all possible splits in all variables

③ Choose the split that reduces impurity the most

④ Each leaf node label is determined by majority decision rule

Pruning:

① Use training data to span Full-Grown Tree

② Choose tree with lowest cost complexity at each pruning stage to prune back

③ Find the point at which the validation error begins to rise

Full-Grown Tree: 100% purity on training (Overfitting!)

Min Error Tree: lowest error rate on validation

Best Pruned Tree: choose smallest with error ⩽ min err + std(min error) (bonus for simplicity)

Cost Complexity: $CC(T) = Err(T) + \alpha L(T)$

$CC(T)$: cost complexity

$Err(T)$: proportion of misclassification

α: penalty factor of tree size

$L(T)$: tree size

Regression Trees:

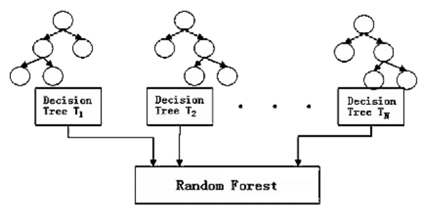
	Classification Trees	Regression Trees
Prediction	Majority Decision Rule	Average
Impurity Measure	Gini Index Entropy	SSE
Performance measure	Error Rate	RMSE

Random Forests:

Take a random sample of training sample size with replacement from training data

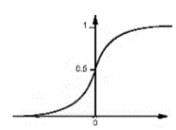
Take a random sample without replacement of the predictors

Construct a tree



Logistic Regression:

Sigmoid function: $g(z) = \frac{1}{1+e^{-z}}$



$odds = e^{\boldsymbol{\beta}^T \boldsymbol{x}}$ Decision Boundary: $\boldsymbol{\beta}^T \boldsymbol{x} \geq \boldsymbol{0}$

$$prob = \frac{odds}{1 + odds} = \frac{1}{1 + e^{-\boldsymbol{\beta}^T \boldsymbol{x}}}$$

Cutoff Value: Initial choice is 0.5

If estimated prob. > cutoff, classify as "1"

Cost Function:

$$J(\boldsymbol{\beta}) = -\frac{1}{m}[(\boldsymbol{y}^T \log \boldsymbol{h} + (\boldsymbol{1} - \boldsymbol{y})^T \log(1 - \boldsymbol{h})]$$

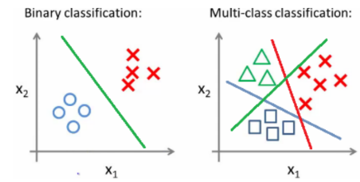
Gradient Descent:

for i = 1 to m

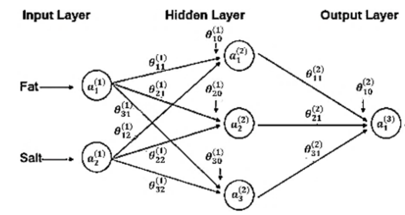
$$\beta_j := \beta_j - \frac{\alpha}{m}(h^{(i)} - y^{(i)})x_j^{(i)}$$

Multiclass Classification: one-vs-all

Samples of one class are positives and all other samples are negatives.



Neural Nets:



Transfer Function:

Linear: $g(x) = \boldsymbol{\beta}^T \boldsymbol{x}$

Exponential: $g(x) = e^{\boldsymbol{\beta}^T \boldsymbol{x}}$

Sigmoid: $g(x) = \frac{1}{1+e^{-\boldsymbol{\beta}^T \boldsymbol{x}}}$

Bipolar Sigmoid: $g(x) = \frac{1-e^{-\boldsymbol{\beta}^T \boldsymbol{x}}}{1+e^{-\boldsymbol{\beta}^T \boldsymbol{x}}}$

Data Processing:

① Normalize to [0, 1]

② Categorical: equidistance in [0, 1], dummy

③ transform skewed: e.g. log right, power left

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \boldsymbol{\Theta}^{(i)} = \begin{bmatrix} \theta_{10}^{(i)} & \theta_{11}^{(i)} & \theta_{12}^{(i)} & \theta_{13}^{(i)} \\ \theta_{20}^{(i)} & \theta_{21}^{(i)} & \theta_{22}^{(i)} & \theta_{23}^{(i)} \\ \theta_{30}^{(i)} & \theta_{31}^{(i)} & \theta_{32}^{(i)} & \theta_{33}^{(i)} \end{bmatrix}$$

Forward Propagation:

$\boldsymbol{a}^{(1)} = \boldsymbol{x}$ (add x_0) $\quad \boldsymbol{z}^{(2)} = \boldsymbol{\Theta}^{(1)} \boldsymbol{a}^{(1)}$

$\boldsymbol{a}^{(2)} = g(\boldsymbol{z}^{(2)})$ (add $a_0^{(2)}$) $\quad \boldsymbol{z}^{(3)} = \boldsymbol{\Theta}^{(2)} \boldsymbol{a}^{(2)}$

$\boldsymbol{h} = \boldsymbol{a}^{(3)} = g(\boldsymbol{z}^{(3)})$

Back Propagation

$\boldsymbol{\delta}^{(3)} = \boldsymbol{h} - \boldsymbol{y}$

$\boldsymbol{\delta}^{(2)} = {\boldsymbol{\Theta}^{(2)}}^{\boldsymbol{T}} \boldsymbol{\delta}^{(3)} .* \, g'(\boldsymbol{z}^{(2)})$

$g'(\boldsymbol{z}^{(2)}) = \boldsymbol{a}^{(3)} * (\boldsymbol{1} - \boldsymbol{a}^{(3)})$

$\boldsymbol{a}^{(1)}$ is input, no error

Cost Function: Same as Logistic Regression

Gradient Descent:

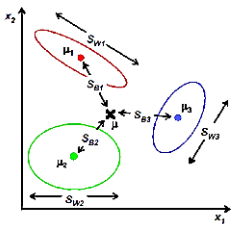
for i = 1 to m

$\theta_{ji}^{(l)} := \theta_{ji}^{(l)} - \alpha \frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{ji}^{(l)}} \quad \frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_{ji}^{(l)}} = \frac{1}{m} a_i^{(l)} \delta_j^{(l+1)}$

Linear Discriminant Analysis:

To classify a new record, measure its distance from the center of each class

Centroid：The centroid is the center of class, which is the mean vector of all records that belong to the class



Characteristics:

Classification, Model-based, Classical statistical, Suitable for small datasets

Assumptions:

equal correlations within each class, and normality

Classification Functions:

Fisher's Linear:

$$\hat{s}_k\left(\boldsymbol{x}^{(i)}\right) = -\frac{1}{2}\bar{\boldsymbol{x}}_k{}^{\boldsymbol{T}}\boldsymbol{S}^{-1}\bar{\boldsymbol{x}}_k + \bar{\boldsymbol{x}}_k{}^{\boldsymbol{T}}\boldsymbol{S}^{-1}\boldsymbol{x}^{(i)} + lnP_k$$

$\bar{\boldsymbol{x}}_k$ is centroid of class k

P_k is prior probability of class k

$+lnP_k$ is to deal with inequity frequency

Other functions: e.g. $\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}$

Steps:

① Measure each record distance from centroids

Euclidean Distance, <u>Statistical Distance</u>

② Classify a record to class with highest score of classification functions

③* Converting to probabilities and then compare to the cutoff value

$$P\left(\boldsymbol{x}^{(i)} \in C_k\right) = \frac{e^{\hat{s}_k\left(\boldsymbol{x}^{(i)}\right)}}{\sum_{t=1}^{c} e^{\hat{s}_t\left(\boldsymbol{x}^{(i)}\right)}}$$

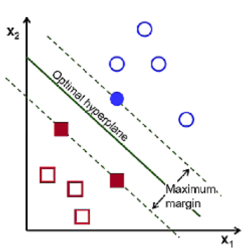
Unequal Misclassification Costs:

Add log $Cost$ or Add log $\frac{Cost_2}{Cost_1}$ to class 2's score

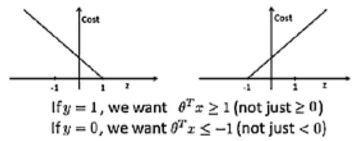
Support Vector Machine:

Create a hyperplane which divides the space to gain homogeneous partitions on either side

Classification & Numerical



Cost Function:



$$J(\boldsymbol{\theta}) = -C\sum_{i=1}^{m}\left(y^{(i)}\text{Cost}_1 + (1-y^{(i)})\text{Cost}_0\right) + \frac{1}{2}\sum_{j=1}^{p}\theta_j{}^2$$

Decision Boundary to $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$:

$p^{(i)}$ is length of projection of $\boldsymbol{x}^{(i)}$ on $\boldsymbol{\theta}$

$$\begin{array}{l}\min_{\boldsymbol{\theta}} \frac{1}{2}\sum_{j=1}^{p}\theta_j{}^2\\ \text{s.t. } \boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}^{(i)} \geq 1\\ \quad \text{if } y^{(i)} > 0;\\ \quad \boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x}^{(i)} \leq 1\\ \quad \text{if } y^{(i)} < 0\end{array} \quad \rightarrow \quad \begin{array}{l}\min_{\boldsymbol{\theta}} \|\boldsymbol{\theta}\|^2\\ \text{s.t. } p^{(i)}\|\boldsymbol{\theta}\| \geq 1\\ \quad \text{if } y^{(i)} > 0;\\ \quad p^{(i)}\|\boldsymbol{\theta}\| \leq 1\\ \quad \text{if } y^{(i)} < 0\end{array}$$

Kernel for Non-linear:

Linear: $K\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right) = \boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)}$

Polynomial: $K\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right) = \left(\boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)} + 1\right)^{\boldsymbol{d}}$

Sigmoid: $K\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right) = tanh\left(\kappa\boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)} - \delta\right)$

Linear: $K\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right) = e^{-\frac{\left\|x^{(i)} - x^{(j)}\right\|^2}{2\sigma^2}}$

Association Rules:

Rules:

If A(Antecedent), then C(Consequent)

Item Sets: combination of items

$Support(A) = P(A) = \frac{\# A}{N}$

Apriori Algorithm:

Set a minimum support criterion → generate list of one-item sets → Use the list of one-item sets to generate list of two-item sets → … → Stop when no k-item sets satisfy criterion

Performance Measure:

Confidence shows the rate at which consequents will be found

$Cofidence = P(Consequent|Antecedent)$

$Cofidence = \frac{P(Consequent\ and\ Antecedent)}{P(Antecedent)}$

$Cofidence(X \rightarrow Y) = \frac{Support(X,Y)}{Support(X)} = \frac{\# X\&Y}{\# X}$

$Cofidence(Y \rightarrow X) = \frac{Support(X,Y)}{Support(Y)} = \frac{\# X\&Y}{\# Y}$

Lift Ratio:

Lift ratio shows how efficient the rule is in finding consequents

$Lift\ Ratio = \frac{Cofidence}{Benchmark\ Cofidence}$

$Benchmark\ Cofidence = P(Consequent)$

Collaborative Filtering:

Find <u>best</u> no-rated item from k nearest neighbors

Similarity Measures:

① User Correlation (Only co-rated items):

$Cor(U_1, U_2) = \frac{\sum(r_{1i} - \bar{r}_1)(r_{2i} - \bar{r}_2)}{\sqrt{\sum(r_{1i} - \bar{r}_1)^2 \sum(r_{2i} - \bar{r}_2)^2}}$

The average ratings for each user are across all items

② Cosine Similarity Measure:

$Cos\ sim(U_1, U_2) = \frac{\sum r_{1i} r_{2i}}{\sqrt{\sum r_{1i}{}^2 \sum r_{2i}{}^2}}$

Suggest normalizing rate first

③ Jaccard Similarity Measure (for binary data):

$J(U_1, U_2) = \frac{U_1 \cap U_2}{U_1 \cup U_2}$

k-Means Cluster Analysis:

Dividing data into k clusters or groups automatically

Steps:

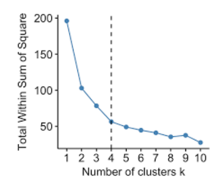
① Randomly initiate cluster centroids $\boldsymbol{\mu}_{\boldsymbol{t}}$

② for each instance i, $\min_t \left\|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_{\boldsymbol{t}}\right\|$

③ Recalculate centroids, $\boldsymbol{\mu}_{\boldsymbol{t}} = E\left[\boldsymbol{x}^{(i)} | \boldsymbol{x}^{(i)} \in C_t\right]$

④ Repeat ② ③

⑤ Elbow Method in cost function to choose k



Cost function: $J(\boldsymbol{C}, \boldsymbol{\mu}) = \frac{1}{m}\sum_{i=1}^{m}\left\|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_{\boldsymbol{C}_{(i)}}\right\|^2$

Hierarchical methods:

①Agglomerative methods: start, then merge

②Divisive methods: divide one cluster again

Similarity Measure:

Distance, Correlation (Numerical)

Matching Coef.: $\frac{TP + TN}{TP + FP + FN + TN}$ (Categorical)

Jaccard's Coef.: $\frac{TP}{TP + FP + FN}$ (Categorical)

Gower's: $S_{ij} = \frac{\sum_{k=1}^{p} w_{ijk} S_{ijk}}{\sum_{k=1}^{p} w_{ijk}}$ (Mixed)

For numerical:

$S_{ijk} = 1 - \frac{\left|x_k^{(i)} - x_k^{(j)}\right|}{Max(x_k) - Min(x_k)} \quad w_{ijk} = 1$

For categorical:

$x_k^{(i)}$	+	+	-	-
$x_k^{(j)}$	+	-	+	-
S_{ijk}	1	0	0	0
w_{ijk}	1	1	1	0